A GEOMETRIC D/A CONVERTER FOR A DELAY-LOCKED LOOP

Related Applications

This is a continuation of Application Serial No. 10/396,884 filed March 25, 2003. This application is incorporated herein by reference.

Background of the Invention

1. Field of the Invention

This invention relates generally to digital-to-analog converters (DACs), and more particularly to a geometric D/A converter architecture that can be used to implement a delay-locked loop (DLL) with a digital control loop.

2. Description of the Prior Art

The general architecture of a DLL 100 with a digital control loop is shown in Figure 1. The DLL 100 includes of a delay line 102 with controllable delay (shown as a chain of buffers in the Figure). The control signal 104 is analog. DLL 100 further includes a phase detector 106 (shown as pd) which compares the input and output clock signals 108, 110 of the delay line 102 and issues an appropriate signal in response to a mismatch. The DLL 100 also includes a filter and state machine 112 which receives the phase detector results and makes an appropriate decision with regard to any delay increase or decrease. Finally, DLL 100 can also be seen to include a D/A converter 120 (shown and referred to herein after as DAC), which converts the filter and state machine 112 results from digital format to analog, which are then fed to the delay line control line 104.

The cells of delay line 102 can be, for example, current starved inverters, meaning that the delay of each cell is controlled by varying the amount of current available for switching. In this implementation, the output of the DAC 120 is a current. The delay of a current starved inverter is known to be proportional to C·Vdd/I, where C is the input capacitance of the gate, Vdd is the supply voltage, and I is the current supplied to the cell.

If a linear DAC 120 is employed along with the delay line 102 to provide a 1% delay resolution, for example, the 1% delay resolution should be set at the minimum current or maximum delay. Then, if for example, a delay range of 2-4nsec is desired and if the associated process, voltage and temperature (PVT) variations and margins are assumed to be 65%, the delay resolution will be 24-40psec at 4nsec (0.6%-1%) and 6-10psec at 2nsec (0.3%-0.5%). The foregoing characteristics necessitate at least 230 taps for the DAC 120 (number of taps = $(I_{max}/I_{min}-1)./\ 0.01 = (2\times1.65-1)/0.01 = 230$). Considering the unnecessary fine resolution at 2nsec, this will of course result in wasted resources.

A better solution might be to implement a geometric DAC 120, in which the current output from tap to tap increases geometrically. In the present document, the required resolution is denoted by the term (k-1). For example a resolution of 1% means that k-1=0.01 or k=1.01. Then choosing:

$$I_0 = I_{\min},\ i_1 = I_0(k-1),\ i_2 = I_0(k-1)k,\ i_3 = I_0(k-1)k^2,\ldots$$
$$i_n = I_0(k-1)k^{n-1}$$

renders $I_n = I_0 k^n$, and

$$\text{Delay} \propto 1/(I_0 k^n) \propto (1/k^n).$$

All of the delay steps will thus be k times apart from one another. The same 1% resolution discussed herein before can therefore now be achieved using only 120 taps.

Such a geometric DAC can be implemented simply by using a series of 120 consecutive geometrically sized transistors having 120 output switches. This type of implementation is problematic however, in that the size of the DAC 120 can become

large; and in fact can be as large as the linear DAC discussed herein before. Since 120 control lines will be necessary, an appropriate shift register can be employed, but will also be problematic since it will be large and difficult to control.

The desired geometric DAC can also be implemented by summing the current from a pair of DACs. One DAC would be a sub-DAC with n output taps, each one providing the minimum current resolution. The primary DAC would have m output taps, with each tap providing the current resolution of n taps combined (k^n-1 for a geometric DAC), where m should be (total number of taps needed)/n. While this technique is easy to implement using a linear DAC, it does not render itself easily to the implementation of a geometric DAC. Considering errors in transistor sizes, this implementation can even lead to a DAC having an undesirable non-monotonic output current. Further, the overall DAC size remains undesirably the same.

It would therefore be desirable and advantageous in view of the foregoing to provide a geometric D/A architecture suitable for implementing a delay-locked loop with a digital control loop and that does not suffer the shortcomings discussed herein above.

Summary of the Invention

The present invention is directed to a geometric DAC architecture. The DAC architecture includes a series of substantially identical sub-DACs, each sub-DAC having n taps. The sub-DACs are fed from a bias-DAC having m = (total number of taps needed)/n taps. The output of each of the m taps is increased geometrically at a rate of k^n, where k has the same meaning as the same symbol used in the background section. The geometric DAC architecture control lines desirably require only (m + n) bits compared with (m × n) bits for the simpler more conventional approach. Further, the geometric DAC architecture requires less real estate than the simpler more conventional approach, is easy to expand because it is modular, and generates an output current that is always monotonic, regardless of errors in transistor sizes and PVT variations.

According to one embodiment, a digital-to-analog converter (DAC) comprises a series of substantially identical sub-DACs, each sub-DAC having n taps and (n+m) control lines; and a bias-DAC having m output taps, wherein the output of each of the m taps is increased geometrically from its immediately preceding bias-DAC output tap, and further wherein each bias-DAC output tap is configured to bias one sub-DAC.

The DAC may also comprise means for alternately inverting n control lines between sub-DACs such that any state transition associated with the DAC occurs in response to no more than a single bit change in any of the n and m control lines.

According to another embodiment, a method of controlling a geometric DAC comprises the steps of providing a geometric DAC having a plurality of substantially identical sub-DACs, a bias-DAC, and n+m control lines, wherein n=number of sub-DAC outputs and m=number of bias-DAC outputs; coding the m control lines to control which sub-DACs are active; and coding the n control lines to control which sub-DAC taps are active in association with the last known active sub-DAC.

The method of controlling a geometric DAC may also comprise of alternately inverting n control line inputs between sub-DACs such that any state transition associated with the geometric DAC occurs with only one bit change in each of the control lines.

Brief Description of the Drawings

Other aspects and features of the present invention and many of the attendant advantages of the present invention will be readily appreciated as the same become better understood by reference to the following detailed description when considered in connection with the accompanying drawings wherein:

Figure 1 illustrates the general architecture of a delay-locked loop (DLL) with a digital control loop;

Figure 2 is a block diagram illustrating a DAC according to one embodiment of the present invention; and

Figures 3A-3C show a full code listing including the normalized current outputs for the DAC depicted in Figure 2.

Fig. 4 is a diagram illustrating the use of shift registers to implement a coding scheme of the present invention.

While the above-identified drawing figures set forth particular embodiments, other embodiments of the present invention are also contemplated, as noted in the discussion. In all cases, this disclosure presents illustrated embodiments of the present invention by way of representation and not limitation. Numerous other modifications and embodiments can be devised by those skilled in the art which fall within the scope and spirit of the principles of this invention.

Detailed Description of the Preferred Embodiments

Figure 2 is a block diagram illustrating a DAC 200 according to one embodiment of the present invention. DAC 200 employs a geometric architecture. This DAC architecture includes a series of substantially identical sub-DACs 202, each sub-DAC having n taps, where n=10 for this embodiment. The sub-DACs 202 are fed from a bias-DAC 204 having m = (total number of taps needed)/n taps, where m=12 for this embodiment. The output of each of these m taps is increased geometrically at a rate of k^n. The outputs of these sub-DACs, either directly or transformed, are used as the control signal of a delay line. The input current to the bias-DAC 204 is a bias DC current. In the present embodiment, it comes from a bandgap bias generator. (In the illustration 200, the directions of the arrows show the directions of the current flow. For example, the arrows from bias-DAC 204 to sub-DACs 202 are shown to go from sub-DACs 202 to bias-DAC 204, although these currents are the outputs of bias-DAC 204 to sub-DAC 202.)

The control lines 206 will be only m+n bits, compared with mxn=(total number of taps needed) for the simple implementation discussed herein before using 120 consecutive geometrically sized transistors with 120 output switches. Using the exemplary numbers discussed above, the DAC architecture shown in Figure 2 will then require only 22 bits compared to the 120 bits required by the simple implementation. It is easily appreciated this DAC architecture allows the use of simpler and smaller logic implementations.

Those familiar with the DLL art will recognize that a fast mode is generally used in the initial cycles when starting a DLL. The DAC architecture depicted in Figure 2 with reference to the inventive embodiments discussed herein easily allows implementation of such a fast mode. In the fast mode, the DAC 200 current resolution will be relaxed to allow for a faster approach to the requisite locking point. This is easily achieved using the DAC 200 simply by switching the higher order control bits (depicted as Qh in Figure 2) and basically jumping from one sub-DAC to the adjacent one within the plurality of sub-DACs 202.

The present inventors found the DAC architecture illustrated in Figure 2 to provide numerous advantages when compared with known architectures. DAC 200, for example, requires a smaller total transistor area (For the embodiment described herein, the area is only about 2/3 that required using known architectures.). Further, the DAC 200 architecture was found to be extremely modular since the sub-DACs 202 are substantially identical and can be repeated as many times as necessary. The DAC 200 architecture is therefore easily expandable. Another advantage found by the present inventors relates to the DAC 200 output current that is always monotonic, regardless of errors in transistor sizes and PVT variations.

The DAC 200 control bits 206 comprise only m+n bits, in which m is the number of sub-DACs 202 and n is the number of taps from each sub-DAC (12+10 for the embodiment described herein), as stated herein before. The Qh bits shown in Figure 2 control which sub-DACs 202 are active. The Ql bits control which taps in the last active

sub-DAC are on (all other previous sub-DACs are completely on). It can be appreciated that one way of coding the DAC 200, for example, is just to use normal control bits, out of two shift registers. An explanation of this approach is demonstrated for the following example code:

Qh<11:0>:000000000011
Ql<9:0>:0000000001

The foregoing Qh bits signify that sub-DACs 0 and 1 are totally on, and sub-DAC 2 is being controlled by the Ql bits. All other sub-DACs are off. The foregoing Ql bits signify that the first tap out of sub-DAC 2 is on.

Using this normal control bit approach will create glitches at the output of the DAC 200 for certain transitions. Using the embodiment shown in Figure 2, for example, one transition is described as follows:

Qh<11:0>	Ql<9:0>	
000000000011	0111111111	
↓	↓	
000000000111	0000000000	

Since the Qh<11:0> and Ql<9:0> transitions will not occur exactly at the same time, one of the following paths will occur for the transition:

Qh<11:0>	Ql<9:0>	
000000000011	0111111111	
↓	↓	
000000000111	*0111111111*	*(Undesirable Transition State)*
↓	↓	
00000000111	0000000000	

Qh<11:0>	Ql<9:0>	
000000000011	0111111111	
↓	↓	
000000000011	*0000000000*	*(Undesirable Transition State)*
↓	↓	
00000000111	0000000000	

In either case, the DAC 200 will jump to a state where the corresponding output is off from the desirable output by k^{n-1} (k=1.01, n=10 and thus $k^{n-1} \cong 9.4\%$ for the embodiment described herein). This is problematic since too many bits are transitioning for only one state transition.

The present inventors, recognizing the foregoing operational problem, implemented a method of operating the DAC 200 as now described herein below with reference to Figures 3A-3C. Figures 3A-3C, in combination, show a listing of the full code and the normalized current outputs associated with the DAC 200 shown in Figure 2.

Looking first again at Figure 2, the Ql<9:0> code for the sub-DACs 202 was inverted for every other sub-DAC as depicted via inverters 208. The above described transition without use of this inverting technique will now appear as described below using the inverters 208.

Qh<11:0>	Ql<9:0>	
000000000011	0111111111	(Ql controls non-inverted sub-DAC, number 2)
↓	↓	
000000000111	1111111111	(Ql controls inverted sub-DAC, number 3)

The state transition, as can be easily seen, now occurs with only one bit change in each of the control bits. Since the next sub-DAC (number 3) inverts the Ql bits, the 1111111111 code will be identical with the 0000000000 code associated with the previous non-inverted coding. The present inventors considered all the transition times

and the dynamics of the DAC 200 and concluded that it was more appropriate for the Qh bits to transition earlier. This technique was found by the present inventors to negate glitches appearing at the output of the DAC 200. Figures 3A-3C illustrate the full code associated with the foregoing example described with reference to Figure 2.

The inverted coding scheme was found easy to implement using shift registers 400, shown in Figure 4. Qh is implemented with a simple shift register 402. Regarding Ql, the shift register 404 is fed with the inverted version of the last bit.

When DAC 200 current is being increased, the bits in Ql shift register 404 are shifted in 406 direction and the shift register is fed with the inverted version of Ql<9> (due to the presence of an inverter 408). When DAC 200 current is being decreased, the bits are shifted in 410 direction and the shift register is fed with the inverted version of Ql<0> (due to the presence of an inverter 412). The output of this shift register 414 goes to the Ql bits of DAC 200.

Once Ql shift register 404 goes through one full shift, Qh shift register 402 needs to be shifted. When DAC 200 current is being increased, Qh shift happens when Ql<9> and Ql<8> are opposite of each other (ref. Figures 3A-3C). The bits in Qh shift register 402 are shifted in 406 direction and the shift register is fed with a value of "1" (shown as reference numeral 416). When DAC 200 current is being decreased, Qh shift happens when Ql<9> and Ql<0> are the same. The bits in Qh shift register 402 are shifted in 410 direction and the shift register is fed with a value of "0" (shown as reference numeral 418). The output of this shift register 420 goes to the Qh bits of DAC 200.

With reference to the example discussed herein before, Ql has 10 bits (Ql<9:0>). When increasing DAC 200 current, Ql<9> is inverted and fed back to the input of the shift register. Thus, after

Ql<9:0>:0111111111,

the result will be:

Ql<9:0>:1111111111

followed by:

Ql<9:0>:1111111110.

Also, after
Ql<9:0>:1000000000,

the result will be:

Ql<9:0>:0000000000

followed by:

Ql<9:0>:0000000001.

In summary, the coding for high order bits (Qh) is a regular thermometer code. For low order bits (Ql), it is either a regular thermometer code or an inverted thermometer code, depending on Qh.

In more mathematical terms, if there are m sub-DACs (m bias-DAC outputs) and n internal taps for each sub-DAC (m=12, n=10 for this example), then there are Qh<m-1:0> and Ql<n-1:0> digital values, and the total number of individual cases is (m*n). If p denotes the code number of the current output (where $p = 0 \rightarrow (m*n)$), then for code p:

Qh<i> = 1 if i <= int(p/n);
Qh<i> = 0 if i > int(p/n);
Ql<i> = 1 - (int(p/n) mod 2) if i < (p mod n); and

Ql<i> = (int(p/n) mod 2) if i >= (p mod n);

where int() denotes the integer function and mod denotes the modulo function. Note that Ql can also have the inverted form of the above equations.

In summary explanation, a geometric DAC architecture has been described that includes a series of substantially identical sub-DACs, each sub-DAC having n taps. The sub-DACs are fed from a bias-DAC has m = (total number of taps needed)/n taps. The output of each of the m taps is increased geometrically at a rate of k^n. The geometric DAC architecture control lines desirably require only (m + n) bits compared with (m x n) bits for the simpler more conventional approach. A control bit technique has also been described to substantially eradicate glitches at the output of the DAC 200 for certain transitions.

This invention has been described in considerable detail in order to provide those skilled in the DAC art with the information needed to apply the novel principles and to construct and use such specialized components as are required. In view of the foregoing descriptions, it should be apparent that the present invention represents a significant departure from the prior art in construction and operation. However, while particular embodiments of the present invention have been described herein in detail, it is to be understood that various alterations, modifications and substitutions can be made therein without departing in any way from the spirit and scope of the present invention, as defined in the claims which follow. For example, while certain embodiments set forth herein illustrate various hardware implementations, the present invention shall be understood to also parallel structures and methods using software implementations as set forth in the claims.